Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tina Avbelj

Analiza infrardečih spektrov z globokimi nevronskimi mrežami

diplomsko delo

interdisciplinarni univerzitetni
študijski program prve stopnje
računalništvo in matematika

Mentor: izr. prof. dr. Janez Demšar

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Infrardeči spektri, ki jih zajemamo v sinhrotronskih pospeševalnikih, predstavljajo zelo uporabno metodo za določanje kemijske sestave različnih vrst vzorcev, od tkiv do različnih materialov. Analiza teh spektrov navadno zahteva strokovnjake, ki prepoznavajo molekule, ki pripadajo opaženim kombinacijam vrhov v spektru. Dosedanji poskusi avtomatizacije takšne analize s strojnim učenjem temeljijo predvsem na uporabi metode podpornih vektorjev.

Raziščite uporabnost globokih nevronskih mrež za analizo infrardečih spektrov.

Zahvaljujem se mentorju izr. prof. dr. Janezu Demšarju in Marku Toplaku za pomoč pri izdelavi diplomskega dela.

Vsem mojim pa hvala za vse.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
CA	classification accuracy	klasifikacijska točnost
SVM	support vector machine	metoda podpornih vektorjev
ANN	artificial neural network	umetna nevronska mreža
CNN	convolutional neural network	konvolucijska nevronska mreža
MLR	multinomial logistic regression	multinomska logistična regresija

Povzetek

Naslov: Analiza infrardečih spektrov z globokimi nevronskimi mrežami

Avtor: Tina Avbelj

Z opazovanjem absorpcijskega spektra, ki ga dobivamo z obsevanjem določenega vzorca, na primer tkiva, z infrardečo svetlobo, lahko dobimo informacijo o kemijski sestavi vzorca. Pri analizi spektrov pogosto uporabljamo klasifikacijske metode, s katerimi lahko določamo sestavo celotnega vzorca ali njegovih delov. Eden od primernih algoritmov za ta namen so umetne nevronske mreže. V diplomskem delu smo ugotavljali, kako uspešne so umetne nevronske mreže za klasifikacijo infrardečih spektrov. Preizkusili smo jih na več naborih podatkov ter jih primerjali z metodo podpornih vektorjev in multinomsko logistično regresijo. Preverjali smo tudi uspešnost konvolucijskih nevronskih mrež. Umetne nevronske mreže so dosegle primerno točnost, vendar niso veliko boljše od metode podpornih vektorjev, ki je, po drugi strani, bistveno hitrejša.

Ključne besede: umetne nevronske mreže, globoko učenje, infrardeči spektri.

Abstract

Title: Analysis of infrared spectra using deep neural networks

Author: Tina Avbelj

Absorption spectra obtained from the sample irradiated by infrared radiation represent a very useful method for observing the chemical composition of different kinds of samples, from cell tissue to various materials. For spectrum analysis, we often use classification algorithms. A suitable algorithm for this task is an artificial neural network. In the diploma thesis, we explored the usefulness of artificial neural networks for classification of infrared spectra. We measured classification accuracies on different data sets and compared them to the results of support vector machines and multinomial logistic regression. We also examined the performance of convolutional neural networks. The results achieved by the artificial neural networks were promising. However, they were not significantly better than those of the support vector machines. On the other hand, the performance of the latter was considerably faster.

Keywords: artificial neural networks, deep learning, infrared spectra.

Poglavje 1

Uvod

Infrardeča spektroskopija je pogosto uporabljena metoda za različne raziskave s področja medicine, kemije, farmacije, forenzike, biologije, ekologije... Infrardeči spekter je pomemben vir informacij o vzorcu, saj iz njega lahko določimo sestavo vzorca. V tovrstnih raziskavah je velik potencial za rešitev marsikaterih problemov. Na podlagi spektrov se da napovedati, ali je neka celica zdrava ali ne in diagnosticirati bolezen. Veliko raziskav te vrste poteka na primer v povezavi z rakom, uporabljajo se za raziskovanje tkiv rastlin in živali, vsebnosti ogljikovega dioksida v toplih gredah, določanje lastnosti materialov in še marsikaj drugega. Probleme na podlagi infrardeče spektroskopije se pogosto rešuje z algoritmi za kalsifikacijo.

V diplomskem delu smo se ukvarjali s klasifikacijo infrardečih spektrov, ki so bili pridobljeni v sinhrotronih *Elettra Sincrotrone Trieste* v Trstu in *SOLEIL* v Parizu. Za to smo uporabili umetne nevronske mreže, ki so zelo razširjen algoritem strojnega učenja. Umetne nevronske mreže se velikokrat izkažejo za zelo uspešen algoritem, saj dajejo visoko točnost napovedi. Cilj diplomskega dela je bil ugotoviti, kakšno klasifikacijsko točnost dobimo z umetnimi nevronskimi mrežami. Njihove rezultate smo primerjali z rezultati multinomske logistične regresije in metode podpornih vektorjev. Ugotavljali smo, kakšna arhitektura umetne nevronske mreže je primerna, in ali se napovedi da izboljšati z uporabo konvolucijske nevronske mreže.

V prvem delu smo predstavili problem klasifikacije in uporabljene algoritme, s poudarkom na umetnih nevronskih mrežah. Sledi opis osnovnega principa delovanja sinhrotrona in razlaga kaj je infrardeči spekter in kako nastane. V naslednjem poglavju je opisan postopek poskusov z različnimi algoritmi in rezultati, sledi pa še zaključek in sklepne ugotovitve.

Poglavje 2

Metode

Za določanje sestave vzorcev s pomočjo spektrov bomo uporabljali strojno učenje, zato v tem poglavju predstavljamo osnovno idejo strojnega učenja ter nekaj najpogostejših algoritmov učenja in z njimi povezanih postopkov. V drugem delu poglavja bomo podrobneje opisali umetne nevronske mreže.

2.1 Klasifikacija

Strojno učenje je področje umetne inteligence, ki se ukvarja z analizo podatkov in podatkovnim rudarjenjem. Osnovni princip strojnega učenja je uporaba algoritma, ki na podlagi podatkov zgradi model, ki ga lahko uporabimo za napovedovanje na novih primerih. Z algoritmi strojnega učenja lahko rešujemo različne vrste problemov, kot je klasifikacija oz. uvrščanje v razrede, napovedovanje vrednosti, razvrščanje v skupine itd.

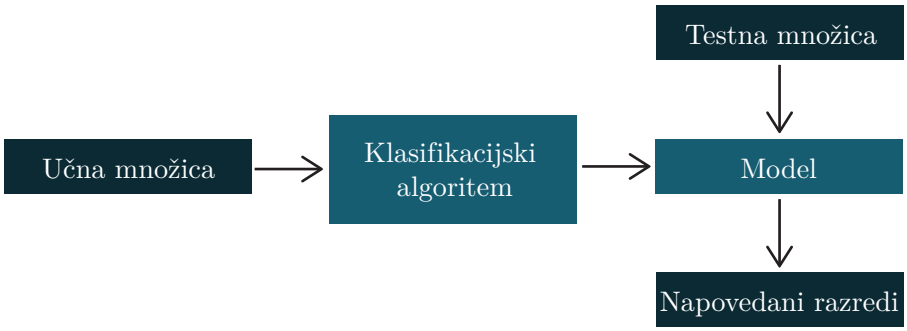
Klasifikacija pomeni, da zgrajeni model za nek primer iz nabora podatkov napove, kateremu izmed možnih razredov pripada. Preprost primer podatkov je množica podatkov o različnih psih, kjer vsak pes pomeni en primer, klasificiramo pa jih v dva razreda: primeren za življenje v stanovanju in neprimeren za življenje v stanovanju. Vsak primer je opisan z atributi (ang. *features*). V prej omenjenem primeru so to velikost, teža, ubogljivost, barva, koliko gibanja potrebuje... Atributi so lahko zvezne (npr. teža) ali diskretne

(npr. barva) spremenljivke. Podatki so predstavljeni kot matrika in vektor. Matrika X predstavlja primere in atribute brez pripadajočega razreda. Vsaka vrstica pomeni en primer, vsak stolpec pa en atribut. [2, 4]

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \tag{2.1}$$

Razredi so predstavljeni z vektorjem y, kjer $y^{(i)}$ pomeni razred za primer oz. vektor $x^{(1)}$.

$$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix} \tag{2.2}$$



Slika 2.1: Shema klasifikacije

2.1.1 Ocenjevanje točnosti in prečno preverjanje

Množico podatkov razdelimo na učno in testno množico. Ponavadi damo v učno množico več primerov, približno v razmerju 7:3. Na učni množici gradimo model. Za primere iz testne množice z zgrajenim modelom napovemo razrede. Točnost klasifikacije merimo s primerjavo pravih in napovedanih

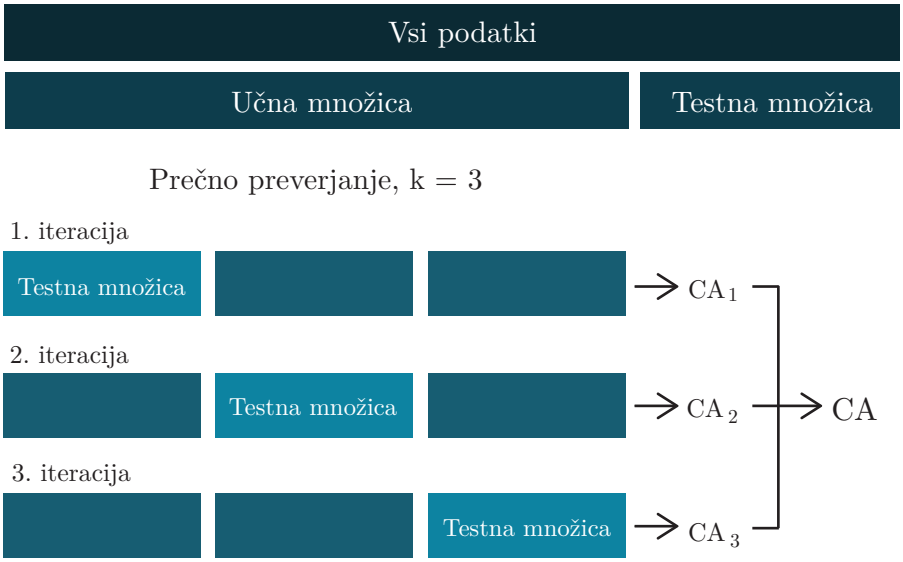
vrednosti razredov za testne primere. Obstaja veliko različnih mer. Ena od njih je klasifikacijska točnost (ang. *classification accuracy*) oz. CA, ki predstavlja razmerje med številom pravilno klasificiranih primerov in številom vseh primerov (n). Če so primeri neenakomerno porazdeljeni v razrede, potem CA ni dobra mera. Obstajajo tudi druge mere, kot so natančnost (ang. *precision*), občutljivost (ang. *recall*), krivulja ROC (ang. *receiver operating characteristic*) itd.

$$CA = 1 - \frac{\sum_{i=1}^{n} |y_{napovedan} - y_{pravilen}|}{n} \tag{2.3}$$

Večina algoritmov strojnega učenja ima parametre, ki vplivajo na točnost metode. Kakšni so optimalni parametri, ne vemo, obstajajo pa algoritmi s katerimi jih lahko najdemo. Dober način za izbiro parametrov je k-kratno interno prečno preverjanje, pri katerem izmerimo točnost za vse kombinacije iz nabora parametrov in končni model zgradimo s tisto, ki da najboljši rezultat. Podatke razdelimo na učno in testno množico. Testne množice pri izbiri parametrov ne uporabljamo in na njej testiramo le končni model. Učno množico razdelimo na k delov. Izvedemo k iteracij. V vsaki izberemo eno od k množic za testno množico, ostalih $k - 1$ pa za učno množico. Vsakič izberemo drugo izmed k množic, tako da je na koncu vsaka od njih natanko enkrat testna množica. Na vsaki izmed k učnih množic zgradimo model z izbranimi parametri in ga testiramo na pripadajoči testni množici. Tako za vsak parameter dobimo k točnosti. Za točnost, ki nam jo da posamezni parameter, vzamemo povprečje vseh k točnosti. S tem zmanjšamo možnost, da bi parametri dali dober rezultat le zaradi naključne izbire množic. Teh k iteracij izvedemo za vsako kombinacijo parametrov, tako da za vsako poznamo točnost. Za gradnjo končnega modela izberemo tiste parametre, ki nam dajo največjo točnost. Končni model testiramo na testni množici, ki smo jo na začetku prečnega preverjanja razdelili od učne. Pričakujemo podoben rezultat, kot ga pri prečnem preverjanju dobimo za najboljšo kombinacijo parametrov.

Prečno preverjanje je lahko za zahtevnejše algoritme preveč dolgotrajno. Hitrejša možnost za izbiro parametrov je validacijska množica. Množico po-

datkov razdelimo na učno, testno in validacijsko množico, na primer v razmerju 6:2:2. Na učni množici zgradimo model, validacijsko množico uporabimo za izbiro najboljših parametrov, na testni množici pa testiramo končni model.



Slika 2.2: Shema internega prečnega preverjanja.

2.1.2 Klasifikacijski algoritmi

Za klasifikacijo se uporabljajo različni algoritmi kot so nevronske mreže, logistična regresija, naključni gozdovi, metoda podpornih vektorjev itd. Nevronske mreže bomo natančneje opisali v naslednjem podpoglavju.

Logistična regresija. Logistična regresija je eden izmed preprostejših klasifikacijskih algoritmov. Navadna logistična regresija deluje samo za primere iz dveh razredov. Za primere iz več razredov pa lahko uporabimo multinomsko logistično regresijo. Algoritem išče hiperravnino, ki primere loči v različna razreda. Parametri algoritma so $\theta_1, ..., \theta_n$. Hipotezo algoritma h

dobimo tako, da izraz z vstavimo v sigmoidno funkcijo g.

$$z = \theta_0 x_0 + \theta_1 x_1 + ... + \theta_n x_n = \theta^T x \tag{2.4}$$

$$g(z) = \frac{1}{1 + exp(-z)} \tag{2.5}$$

$$h_\theta(x) = g(z) = g(\theta^T x) = \frac{1}{1 + exp(-\theta^T x)} \tag{2.6}$$

Izhodna vrednost hipoteze je na intervalu med 0 in 1. Pove nam verjetnost, s katero primer x ob danih parametrih θ pripada pozitivnemu razredu. Če torej primer spada v pozitivni razred, želimo, da je vrednost hipoteze čim bližje 1, če spada v negativni pa 0. Vrednost 0.7 na primer pomeni, da primer spada v pozitivni razred s 70 % verjetnostjo. Algoritem mora izbrati optimalne vrednosti parametrov θ. Za to lahko uporabimo metodo gradientnega spusta (poglavje 2.2.2), ki minimizira kriterijsko funkcijo $J(\theta)$. $y^{(i)}$ lahko zavzame vrednost 1, kar pomeni pozitivni razred, ali 0, kar je negativni razred ($y^{(i)} \in \{0, 1\}$).

$$min\ J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)}\ log(h_\theta\ (x^{(i)}))\ +\ (1 - y^{(i)})\ log(1 - h_\theta\ (x^{(i)}))] \tag{2.7}$$

Za preprečevanje prevelikega prilagajanja učni množici lahko funkciji $J(\theta)$ prištejemo še regularizacijski člen. Poznamo več vrst regularizacije, ki se razlikujejo v uporabljeni normi in različno vplivajo na model. Najbolj pogosti sta L1 in L2. Naslednja enačba predstavlja člen za L2 regularizacijo. [2, 4]

$$\frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2, \tag{2.8}$$

Multinomska logistična regresija. Multinomska logistična regresija (ang. *multinomial logistic regression* oz. *softmax regression*) je generalizacija logistične regresije, s katero lahko klasificiramo primere v več razredov. Če imamo k razredov, lahko $y^{(i)}$ zavzame cele vrednosti od 1 do k ($y^{(i)} \in \{1, 2, ..., k\}$). Za testni primer x torej želimo, da hipoteza oceni verjetnost $P(y = k|x)$, s katero x pripada razredu k, za vse možne vrednosti k. Izhodna vrednost algoritma je k-dimenzionalni vektor y, katerega elementi so

verjetnosti za vsak razred, ki zavzamejo vrednosti med 0 in 1, vsota vseh elementov pa je 1. Izračunamo jo s hipotezo h.

$$h_\theta(x) = \begin{bmatrix} P(y=1|x;\theta) \\ P(y=2|x;\theta) \\ \dots \\ P(y=k|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} 1 + exp(\theta^{(j)T}x)} = \begin{bmatrix} exp(\theta^{(1)T}x) \\ exp(\theta^{(2)T}x) \\ \dots \\ exp(\theta^{(k)T}x) \end{bmatrix} \tag{2.9}$$

Parameter modela θ je $n \times k$ matrika, kjer je n dolžina posameznega vektorja $\theta^{(j)}$.

$$\theta = \begin{bmatrix} \vec{\theta^{(1)}} & \vec{\theta^{(2)}} & \dots & \vec{\theta^{(k)}} \end{bmatrix} \tag{2.10}$$

Kriterijska funkcija pa je naslednja.

$$J(\theta) = -[\sum_{i=1}^{m}\sum_{l=1}^{k} 1\{y^{(i)} = l\} log \frac{exp(\theta^{(l)T}x^{(i)})}{\sum_{j=1}^{k} exp(\theta^{(j)T}x^{(i)})}] \tag{2.11}$$

Z $1\{a\}$ je označena funkcija indikator. Lahko zavzame vrednost 1 ali 0. 1 je v primeru, ko je izraz a pravilen, kar pomeni, da je vrednost napovedanega razreda $y^{(i)}$ enaka k. Sicer ima funkcija vrednost 0. [11]

Metoda podpornih vektorjev. Metoda podpornih vektorjev oz. SVM nam omogoča, da se model lahko nauči tudi nelinearne funkcije. V osnovi je SVM binarni klasifikacijski algoritem, torej za primere iz dveh razredov. Za klasifikacijo v več razredov se problem prevede na binarni z metodo eden proti vsem. To pomeni, da za vsak razred vzamemo za pozitivne primere tiste, ki pripadajo temu razredu, za negativne pa primere iz vseh ostalih razredov. Metoda pri tem zgradi toliko modelov, kolikor je razredov. Pri napovedovanju razreda testnega primera uporabimo vse predhodno zgrajene modele, in sicer tako, da vsak vrne verjetnost izbranega razreda za ta določen primer. Na koncu izberemo tisti razred, ki je bil napovedan z največjo verjetnostjo.

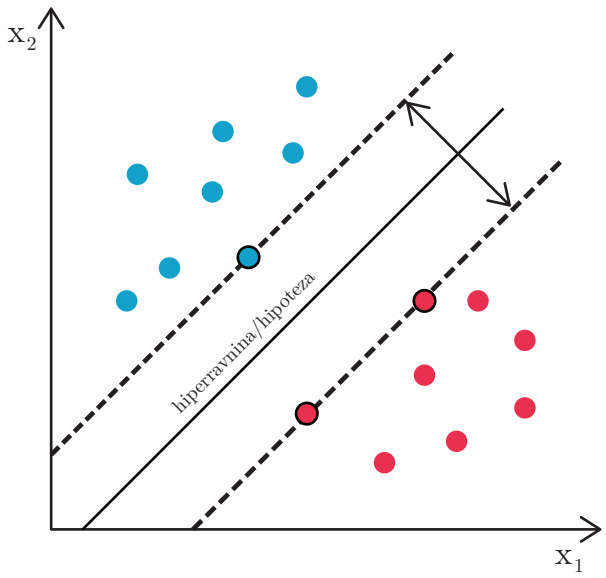
Glavna ideja algoritma je, da loči primere iz različnih razredov s hiperravnino, npr. v 2D prostoru s premico. Primeri oz. vektorji x z n atributi so točke v n dimenzionalnem prostoru. Primere iz različnih razredov algoritem loči s tako hiperravnino h, da je razdalja med njo in najbližjim pozitivnim

primerom ter najbližjim negativnim primerom maksimalna. h je hipoteza algoritma, ki jo lahko zapišemo z naslednjo enačbo.

$$h_\theta(x) = \theta^T x - b = 0 \tag{2.12}$$

Če so primeri linearno ločljivi, lahko izberemo dve hiperravnini, ki ločita primere različnih razredov in sta maksimalno oddaljeni ena od druge, h pa leži na sredini med njima. Največjo razdaljo algoritem določi s pomočjo podpornih vektorjev, ki so tisti učni primeri, ki ležijo blizu ene oz. na eni izmed hiperravnin. Na sliki 2.3 so to primeri, ki so obkroženi s črno.

Parameter algoritma je tudi jedro (ang. *kernel*). Za linearno ločljive primere lahko uporabimo linearno jedro. Vendar primeri niso vedno tako razporejeni, da bi se jih dalo ločiti s hiperravnino. V 2D prostoru bi na primer lahko potrebovali krivuljo. V takih primerih lahko uporabimo drugačno jedro, ki preslika točke v višjedimenzionalni prostor, kjer jih je lažje razdeliti s hiperravnino. Za jedro lahko uporabimo različne funkcije. Najbolj pogosta jedra so linearno, polinomsko in RBF (ang. *radial basis function*). [2, 4]



Slika 2.3: SVM za dva atributa x_1 in x_2.

2.2 Umetne nevronske mreže

V diplomskem delu nas zanima predvsem, kako uporabne so za določanje spektrov umetne nevronske mreže.

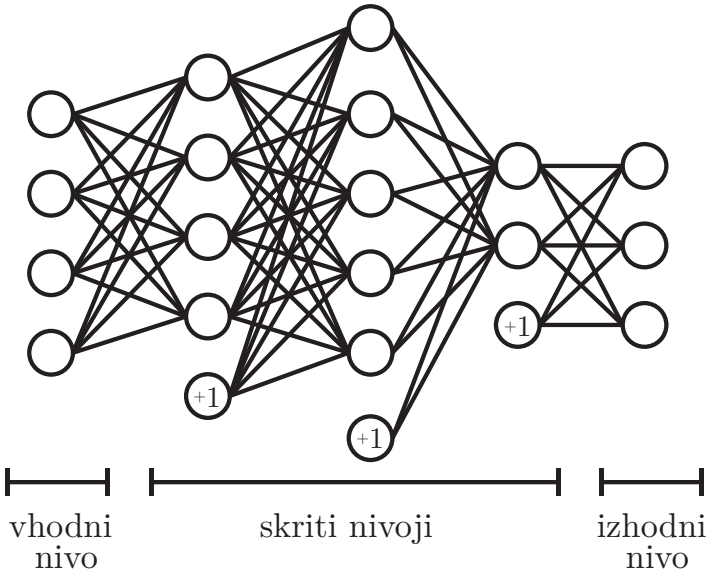
2.2.1 Struktura umetne nevronske mreže

Umetna nevronska mreža je sestavljena iz nevronov. Nevron ima vhodne in izhodne povezave. Preko vhodnih povezav prejme vrednosti, iz katerih izračuna eno izhodno vrednost. Nevroni so povezani v nevronsko mrežo tako, da je izhod enega nevrona vhod drugega nevrona. Umetna nevronska mreža ima več nivojev. Število nevronov na vhodnem nivoju je enako številu atributov. Število nevronov na izhodnem nivoju pa je enako številu različnih razredov, v katere klasificiramo primere. Ostali nivoji se imenujejo skriti nivoji, ki imajo lahko poljubno število nevronov. Število nivojev in nevronov na posameznih nivojih imenujemo arhitektura mreže. Vsak nivo ima tudi dodatni nevron t.i. prag (ang. *bias*), ki je na sliki 2.4 označen s +1. Ta nevron nima vhodnih povezav, njegova izhodna vrednost pa je vedno 1. Nevroni na zaporednih nivojih so med sabo polno povezani, vhodi nekega nevrona so torej izhodi vseh nevronov iz prejšnjega nivoja.

Parametra umetne nevronske mreže sta W in b. W je večdimenzionalna matrika uteži, kjer $W_{ij}^{(l)}$ pomeni utež povezave med nevronom j na nivoju l in nevronom i na nivoju $l+1$. Če ima mreža s_j nevronov na nivoju l in s_{j+1} nevronov na nivoju $j+1$, je dimenzija $W^{(j)}$ enaka $s_{j+1} \times (s_j + 1)$. Parameter b pa se navezuje na prag, kjer $b_i^{(l)}$ pomeni utež povezave med pragom na nivoju l in nevronom i na nivoju $l+1$. Izhodno vrednost oziroma aktivacijo posameznega nevrona označimo z a. $a_i^{(l)}$ pomeni izhod nevrona i na nivoju l. Aktivacijo posameznih nevronov izračunamo z aktivacijsko funkcijo. Lahko izberemo sigmoido (enačba 2.13), ki zavzame vrednosti na intervalu (0,1).

$$f(z) = \frac{1}{1 + e^{-z}} \tag{2.13}$$

Ena od drugih možnosti je hiperbolični tangens (enačba 2.14).



Slika 2.4: Nevronska mreža s tremi skritimi nivoji.

$$f(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.14}$$

Za izračun aktivacije uporabimo še oznako $z_i^{(l)}$, ki pomeni uteženo vsoto vseh vhodov v nevron, vključno s pragom. Aktivacija nevrona i na nivoju l pa je vrednost aktivacijske funkcije v točki $z_i^{(l)}$.

$$z_i^{(l+1)} = \sum_{j=1}^{n} W_{ij}^{(l)} x_j + b_i^{(l)} \tag{2.15}$$

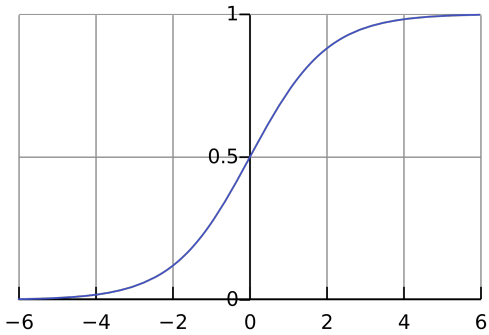
$$a_i^{(l+1)} = f(z_i^{(l+1)}) \tag{2.16}$$

Končni izhod $h_{W,b}(x)$ je enak aktivaciji na zadnjem nivoju.

Ta postopek računanja aktivacij nevronov in končnega izhoda umetne nevronske mreže pri znanih parametrih (W, b) se imenuje usmerjeni algoritem (ang. *feedforward algorithm*). [8, 2, 4, 10]

Primer izračunov aktivacij za preprosto umetno nevronsko mrežo iz slike 2.6:

$$a_1^{(2)} = f(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)}) \tag{2.17}$$

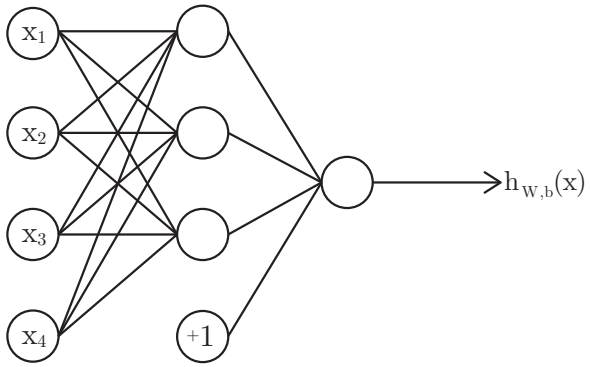


Slika 2.5: Graf sigmoide.

$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \tag{2.18}$$

$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \tag{2.19}$$

$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)}) \tag{2.20}$$



Slika 2.6: Nevronska mreža z enim skritim nivojem.

Pomemben korak pri uporabi umetnih nevronskih mrež je izbira arhitekture mreže. Problem pri tem je, da zanesljivo pravilo, ki bi okvirno opisovalo, kakšno mrežo naj bi izbrali glede na lastnosti podatkov, ne obstaja. Bolj ko je arhitektura mreže kompleksna, večja je zahtevnost računanja in možnost prevelikega prilagajanja učni množici. Zato v splošnem želimo najti čim bolj preprosto oziroma najmanjšo mrežo, ki bi dobro klasificirala podatke. Za to

obstajajo različni pristopi. Ena možnost je preizkušanje različnih arhitektur, kar je pravzaprav ugibanje. To je lahko zelo dolgotrajno in ni nujno, da sploh preizkusimo arhitekturo, ki je dejansko najboljša. Za nekatere probleme se izkaže, da izbira arhitekture sploh ni tako bistvena in lahko dobimo zelo dobre rezultate za veliko različnih mrež, ki vsi presegajo mejo točnosti, ki jo želimo doseči. Obstajajo tudi algoritmični pristopi za izbiro arhitekture. Eden je večanje mreže (ang. *growing neural network*), kjer začnemo z zelo majhnim številom nevronov, nato pa jih dodajamo. Pri rezanju mrež (ang. *pruning neural network*) pa določimo neko zelo veliko mrežo, iz katere odstranjujemo nevrone. [14, 5, 6, 1]

2.2.2 Gradientni spust

Gradientni spust je optimizacijski algoritem, ki se uporablja za različne metode strojnega učenja, med drugim tudi za umetne nevronske mreže. Pri gradientnem spustu je cilj minimizirati kriterijsko funkcijo. Z gradientnim spustom se sicer približujemo le lokalnemu minimumu te funkcije in ne globalnemu, v praksi pa se izkaže, da je to dovolj. V splošnem je ideja algoritma naslednja: izberemo kriterijsko funkcijo $J(\theta_1, \theta_2, ..., \theta_n)$, ki je odvisna od n parametrov, želimo pa najti $minJ(\theta_1, \theta_2, ..., \theta_n)$. Na začetku izberemo naključne parametre $\theta_1, \theta_2, ..., \theta_n$. V vsakem koraku izboljšamo parametre tako, da zmanjšamo $J(\theta_1, \theta_2, ..., \theta_n)$. To ponavljamo, dokler ni vrednost funkcije J dovolj majhna. Nove parametre izberemo glede na odvod oziroma gradient kriterijske funkcije.

V primeru umetnih nevronskih mrež lahko za kriterijsko funkcijo izberemo naslednjo funkcijo.

$$J(W,b) = (\frac{1}{m}\sum_{i=1}^{m}(\frac{1}{2}\|h_{W,b}(x^{(i)}) - y(i))\|^2)) + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(W_{ji}^{(l)})^2 \quad (2.21)$$

V vsaki iteraciji gradientnega spusta izračunamo odvod kriterijske funkcije, s katerim popravimo uteži umetne nevronske mreže (enačbi 2.22 in 2.23). Intuitivno to pomeni, da se premaknemo korak bližje proti minimumu funkcije

J. Parameter α se imenuje stopnja učenja (ang. *learning rate*) in pove, kako velik je korak, s katerim se približujemo minimumu funkcije. Če je α majhen, se približujemo počasi, če je velik, pa hitro. Lahko se zgodi, da izberemo prevelik α in nikoli ne pridemo dovolj blizu minimuma, ker ga zaradi prevelikega koraka v sicer pravi smeri zgrešimo. Pri premajhnem α pa je možno, da se minimumu približujemo prepočasi in se algoritem ustavi, še preden ga dosežemo. Drugi člen kriterijske funkcije s parametrom λ je regularizacijski člen. Regularizacija preprečuje preveliko prilagajanje učnim podatkom. Stopnja regularizacije je določena z λ. Pri večji stopnji je regularizacija večja, torej je prilagajanje učnim podatkom manjše.

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) \tag{2.22}$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b) \tag{2.23}$$

Obstaja več oblik gradientnega spusta, ki se razlikujejo po številu učnih primerov, ki jih upoštevamo v vsaki iteraciji gradientnega spusta.

- Paketni gradientni spust (ang. *batch gradient descent*). Gradient kriterijske funkcije izračunamo iz celotne učne množice. Algoritem torej za vsak popravek uteži uporabi vse primere, zato je lahko počasen. Ob pravilni izbiri stopnje učenja vedno konvergira k lokalnemu minimumu.

- Stohastični gradientni spust (ang. *stochastic gradient descent*). Za vsak učni primer algoritem naredi en popravek uteži. Ta algoritem je hitrejši kot paketni gradientni spust. V praksi se izkaže, da ta algoritem dovolj dobro deluje in skoraj vedno konvergira k lokalnemu minimumu.

- Mini-paketni gradientni spust (ang. *mini-batch gradient descent*). To je kombinacija obeh prejšnjih oblik. Za popravek uteži uporablja podmnožice učne množice z n primeri. Namesto da iteriramo čez vsak primer, kot pri stohastičnem spustu, iteriramo čez podmnožice z n primeri. Za učenje umetnih nevronskih mrež se ponavadi uporablja ta algoritem. [8]

Algoritem vzvratnega razširjanja. Ključni korak v gradientnem spustu je izračun odvoda kriterijske funkcije J. Učinkovit način za računanje odvoda je algoritem vzvratnega razširjanja (ang. *backpropagation algorithm*). Najprej z usmerjenim algoritmom izračunamo vse aktivacije. Na izhodnem nivoju dobimo izračunane aktivacije, poznamo pa tudi pravilno vrednost, zato lahko izračunamo napako. Napako z zadnjih nivojev posredujemo proti prvim. Za vsak testni primer bi torej lahko rekli, da gremo enkrat od vhoda do izhoda mreže in potem enkrat od izhoda proti vhodu. Rezultat algoritma so odvodi, ki jih potrebujemo za izračun gradientnega spusta, s katerim popravljamo uteži W. [8]

1. Z usmerjenim algoritmom izračunamo vse aktivacije za vse nivoje L_1, L_2, ..., L_{n_l} (izhodni nivo).

2. Za vsak nevron na izhodnem nivoju izračunamo napako.
$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \|y - h_{W,b}(x)\|^2 \tag{2.24}$$

3. Vzvratno razširjanje napake po mreži nazaj, s postopnim pomikanjem po nivojih. Za vsak nivo $l = n_l - 1, n_l - 2, ..., 2$ in vsak nevron i na nivoju l izračunamo napako.
$$\delta_i^{(l)} = (\sum_{j=1}^{s_{l+1}} W_{ij}^{(l)} \delta_j^{(l+1)}) f'(z_i^{(l)}) \tag{2.25}$$

4. Izračunamo parcialne odvode kriterijske funkcije.
$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W) = a_j^{(l)} \delta_i^{(l+1)} \tag{2.26}$$
$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W) = a_j^{(l)} \delta_i^{(l+1)} \tag{2.27}$$

2.2.3 Dodatne lastnosti nevronskih mrež

Ko uporabljamo umetno nevronsko mrežo, moramo torej določiti več njenih lastnosti. Najprej izberemo arhitekturo. V primeru, da se odločimo za

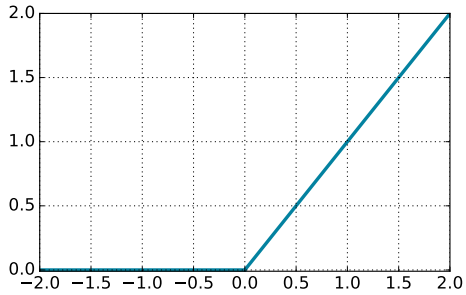
mini-paketni gradientni spust, določimo velikost posameznega mini-paketa. Obstajajo pa tudi druge dodatne možnosti, ki jih lahko uporabimo za izboljšanje delovanja mreže.

Dropout. Pri gradnji modela pogosto pride do prevelikega ujemanja s testnimi podatki (ang. *overfitting*). Eden od načinov reševanja tega problema v primeru umetnih nevronskih mrež je dropout. S to tehniko izpustimo nekatere nevrone v mreži na kateremkoli nivoju. To pomeni, da nevron med učenjem začasno odstranimo iz mreže, vključno z vsemi njegovimi vhodnimi in izhodnimi povezavami. Izbira izpuščenih nevronov je naključna. Za vsak nevron določimo verjetnost, s katero je odstranjen iz mreže. Verjetnost je najbolje izbrati z validacijsko množico ali prečnim preverjanjem, kar pomeni, da preverimo točnost za različne verjetnosti in za gradnjo končnega modela uporabimo najboljšo. [12]

ReLU. ReLU je kratica za angleški izraz *rectified linear unit*. To je funkcija, ki je definirana z naslednjim izrazom.

$$f(x) = max(0, x) \tag{2.28}$$

To funkcijo uporabimo kot aktivacijsko funkcijo. Z x je označen vhod v nevron. Vse vrednosti, ki so manjše od 0, dobijo vrednost 0. [6]



Slika 2.7: Graf funkcije ReLU.

2.2.4 Konvolucijske nevronske mreže

Konvolucijske nevronske mreže so posebna oblika nevronskih mrež, ki upoštevajo tudi sosednje atribute. Uporabimo jih lahko, ko so atributi med sabo korelirani. Tipično se uporabljajo za slike.

Konvolucijske nevronske mreže uporabljajo matematično operacijo konvolucija. To je operacija med funkcijo f in jedrom oz. filtrom g, njen rezultat pa je nova funkcija. Definirana je za zvezne (enačba 2.29) in diskretne (enačba 2.30) funkcije. Označimo jo z znakom $*$.

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(u)g(x-u)du \tag{2.29}$$

$$f(x) * g(x) = \sum_{u=-\infty}^{\infty} f(u)g(x-u) \tag{2.30}$$

Arhitektura konvolucijskih nevronskih mrež je podobna arhitekturi navadnih umetnih nevronskih mrež, le da imajo poleg polno povezanih nivojev tudi en ali več konvolucijskih nivojev. Konvolucijsko jedro si lahko predstavljamo kot uteži. Vsak konvolucijski nivo lahko uporabi drugačno jedro.

Konvolucija je računsko zahtevna, zato lahko uporabimo tehniko združevanje maksimalnih vrednosti (ang. *max pooling*), s katero zmanjšamo število parametrov oz. dimenzijo slike. Ponavadi se združevanje uporabi po ali pred vsakim konvolucijskim nivojem. Deluje tako, da določimo velikost okna, s katerim postopno drsimo čez celoten vhodni primer. Pri vsakem koraku ohranimo le največjo izmed vrednosti, ki se pojavijo v tem oknu. Celotna zgradba konvolucijske nevronske mreže je ponavadi taka, da najprej uporabimo konvolucijske nivoje, med katerimi uporabimo združevanje maksimalnih vrednosti, nato pa enega ali več polnopovezanih nivojev, na katerih lahko uporabimo ReLU in dropout. [3, 6, 1]

2.2.5 Knjižnice za nevronske mreže

Umetnih nevronskih mrež v praksi ponavadi ne programiramo sami, saj za to obstaja veliko knjižnic, pri katerih moramo le definirati arhitekturo mreže

Slika 2.8: Združevanje maksimalnih vrednosti za 4x4 matriko z oknom 2x2.

in so precej preproste za uporabo. Ena izmed bolj razširjenih je Theano, ki je napisana v programskem jeziku Python. Vedno bolj uporabljena je tudi Googlova knjižnica za strojno učenje TensorFlow, ki je prav tako napisana v Pythonu. Obe sta tudi odprtokodni. Še bolj preprosta je knjižnica Keras, ki je namenjena izključno nevronskim mrežam. Uporabljamo jo kot dodatek h knjižnici Theano ali TensorFlow, da lahko kodo pišemo še krajše in bolj pregledno.

Poglavje 3

Podatki

V tem poglavju smo se osredotočili na podatke, njihovo pridobivanje in značilnosti. Podatki so bili pridobljeni v sinhrotronu, ki je predstavljen v nadaljevanju. Sledi opis infrardečih spektrov in oblike podatkov, nato pa so predstavljeni še uporabljeni nabori podatkov.

3.1 Sinhrotron

Podatki so bili pridobljeni v dveh raziskovalnih centrih *Elettra Sincrotrone Trieste* v Trstu in *SOLEIL* v Parizu v sinhrotronu, ki je oblika pospeševalnika delcev. Pospeševalnik je naprava za pospeševanje nabitih delcev na visoko hitrost. Obstaja več vrst pospeševalnikov. V linearnih pospeševalnikih se elektroni gibljejo v ravni črti, v krožnih pa v krogu. Prednost krožnih je, da lahko delce pospešujejo neskončno dolgo, ker se gibljejo v krogu, medtem ko je pot pospeševanja pri linearnih omejena z dolžino pospeševalnika. Krožni pospeševalnik je tudi manjši kot primerljiv linearni, v katerem lahko elektroni dosežejo enako energijo.

Sinhrotron je oblika krožnega pospeševalnika. Pri gibanju z zelo visoko hitrostjo elektroni oddajajo sevanje, imenovano sinhrotronska svetloba. Sevanje je mogoče v pospeševalniku doseči umetno, do njega pa lahko pride tudi naravno, npr. pri različnih astronomskih pojavih. Sinhrotroni sicer

niso vedno namenjeni pridobivanju sinhrotronske svetlobe, vendar se bomo v nadaljevanju osredotočili na te, ker so bili tako pridobljeni naši podatki. Glavni del sinhrotrona je krožna vakuumska cev. V resnici pot elektronov ne poteka v krogu, saj je sinhrotron sestavljen iz ravnih in ukrivljenih delov, ki skupaj tvorijo krožni cikel, ki od zunaj izgleda kot krog. Na ravnih delih se elektroni gibljejo približno enakomerno in ne sevajo. Na ukrivljenih delih pot elektronov usmerjajo z magnetnim poljem, saj bi sicer šli naravnost. Med zavijanjem v magnetnem polju se elektroni gibljejo pospešeno in ob tem oddajo sevanje oz. sinhrotronsko svetlobo. Sevajo naravnost, kar med zavijanjem pomeni v smeri tangente na ovinek. Na ovinkih grejo zato iz pospeševalnika ravne žarkovne linije (ang. *beamline*) v smeri sevanja. Po njih potuje izsevana sinhrotronska svetloba do eksperimentalnih postaj (ang. *endstation*), kjer jo pretvorijo v podatke, s katerimi lahko izvajajo raziskave. V sinhrotronu je tudi več enot, imenovanih resonančne votline, kjer delce pospešujejo z električnim poljem. Pospešijo jih skoraj do svetlobne hitrosti. Ko se hitrost elektronov veča, se mora večati tudi jakost magnetnega polja na ovinkih, da lahko ukrivlja pot hitrejših elektronov. V sinhrotron pošljejo več elektronov naenkrat, ki krožijo v skupinah. Imeti morajo neko začetno hitrost, zato jih je potrebno pred vstopom v glavni krožni pospeševalnik pospešiti. Ob sinhrotronu je ponavadi še en pospeševalnik, ki je lahko linearen ali manjši krožen, kjer se začne pot elektronov, v njem pa dosežejo neko hitrost, s katero jih pošljejo v glavni krožni pospeševalnik. Na koncu elektrone spustijo iz kroga.

Sinhrotronska svetloba ima veliko lastnosti, zaradi katerih je zelo uporabna za raziskave. Spekter sinhrotronske svetlobe je zvezen in gladek, razteza pa se od infrardeče prek vidne in ultravijolične do rentgenske svetlobe. To je koristno, saj lahko izberemo, kateri del spektra bomo preučevali. Najpogosteje je to ultravijolična ali rentgenska svetloba. Snop svetlobe, ki ga odda elektron, je zelo ozek in močen, kar omogoča večjo natančnost. Elektron ne seva konstantno, ampak oddaja sevanje na določene časovne intervale ob zavijanju in za zelo kratek čas. Tudi ta časovna struktura sevanja je koristna

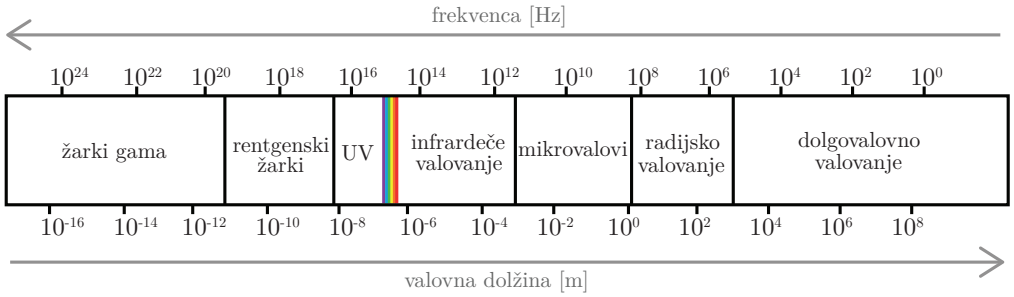
lastnost.

Iz sinhrotronske svetlobe se da ugotoviti marsikaj o sestavi vzorca, zato se uporablja za raziskave na različnih področjih, kot so farmacija, medicina, kemija in biologija. Nekaj primerov raziskav, ki jih delajo s pomočjo spektrov so odkrivanje zgradbe snovi, rentgenska spektrografija proteinov, preučevanje lastnosi materialov, vezi v kemijskih spojinah, tkiv v živih bitjih, raziskave različnih bolezni itd. [7, 13]

3.2 Infrardeči spekter

Infrardeča spektroskopija se ukvarja z infrardečim delom spektra elektromagnetnega valovanja oz. svetlobe. Ključni lastnosti svetlobe, na podlagi katerih lahko določimo, kakšna je svetloba, sta valovna dolžina in frekvenca. Frekvenca pomeni število valov na časovno enoto. Med sabo sta lastnosti odvisni, njuno povezavo pa predstavlja enačba 3.1, kjer je z ν označena frekvenca, z λ valovna dolžina, s c pa hitrost svetlobe, ki je približno $3 \times 10^8 \frac{m}{s}$. Infrardeča svetloba je svetloba z daljšo valovno dolžino in manjšo frekvenco od vidne svetlobe.

$$\nu = \frac{c}{\lambda} \tag{3.1}$$



Slika 3.1: Elektromagnetno valovanje

Infrardeča spektroskopija temelji na tem, da vzorec absorbira določene

frekvence infrardeče svetlobe glede na to, kakšne so lastnosti njene zgradbe. Izračun teh frekvenc je precej zapleten; molekule namreč absorbirajo frekvence, ki so podobne oz. v resonanci s frekvencami vibracij molekulskih vezi. Absorpcijski spekter pove, koliko energije (oz. fotonov) z določeno frekvenco molekula absorbira. Infrardeči spekter vzorca je torej pridobljen tako, da je vzorec izpostavljen infrardečemu sevanju, ki ga oddajo pospešeni elektroni v sinhrotronu, ob tem absorbira svetlobo pri določenih frekvencah, na podlagi tega pa določijo spekter vzorca, iz katerega je možno razbrati, katere molekule vsebuje.

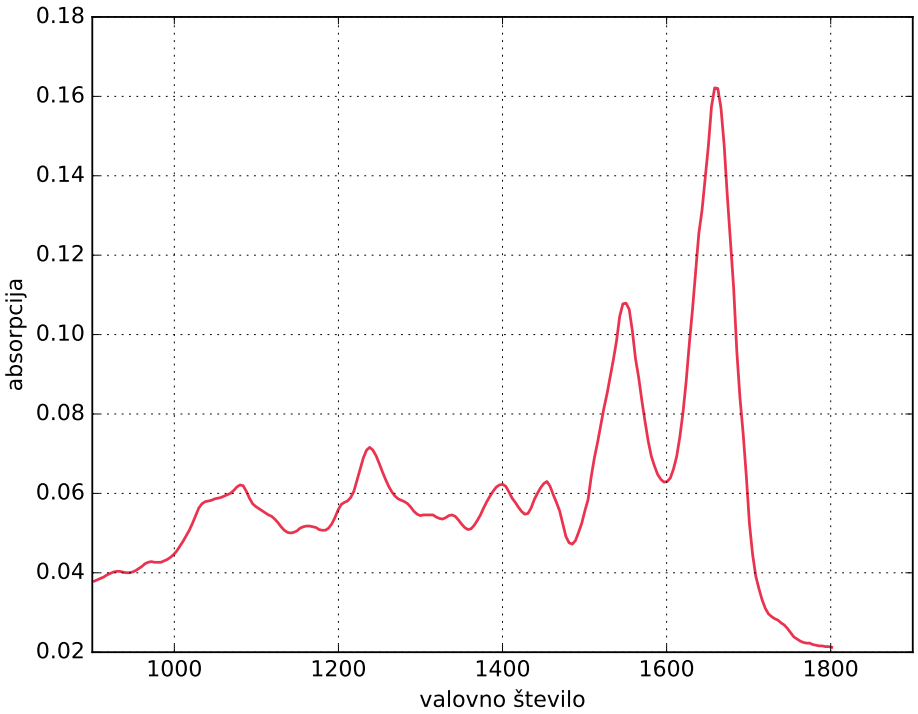
Primer infrardečega spektra iz naših podatkov je predstavljen na sliki 3.2. Na x osi je valovno število (ang. *wavenumber*), ki ga označimo s k, enote pa so cm^{-1}. Valovno število je lastnost valovanja, ki je z valovno dolžino povezano z naslednjo enačbo.

$$k = \frac{2\pi}{\lambda} \tag{3.2}$$

Predstavlja število valov na enoto dolžine v smeri širjenja valovanja. Na y osi pa je absorpcija vzorca oz. kolikšen del izsevane infrardeče svetlobe z določenim valovnim številom je vzorec absorbiral. Za absorpcijo se uporablja več različnih enot, v našem primeru pa ni pomembno, katere so. [9]

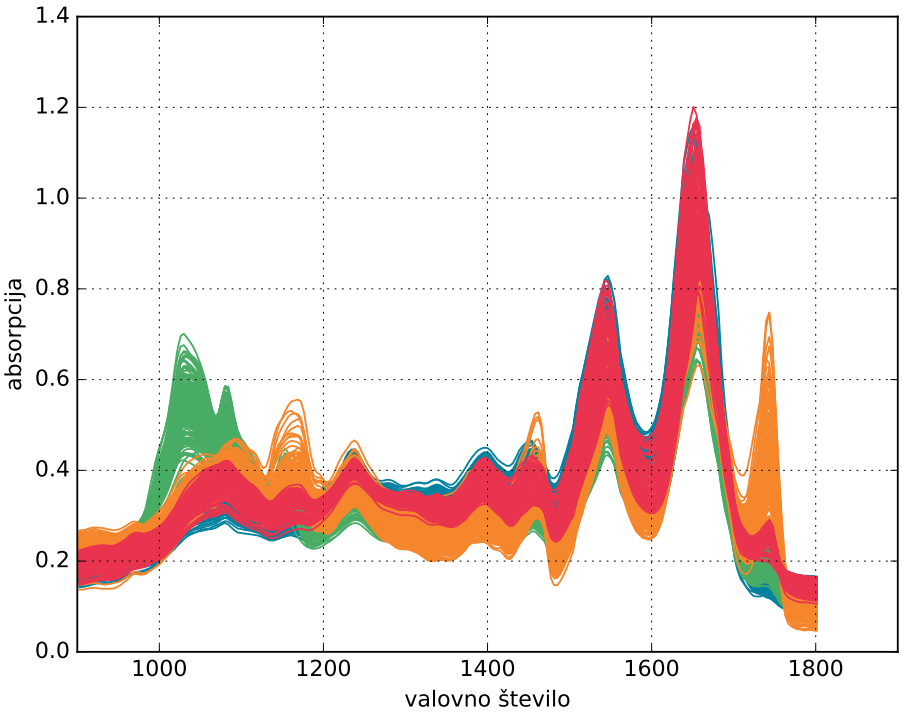
3.3 Nabori podatkov

V podatkih, ki smo jih uporabili, je zajet le infrardeči del spektra. Vsak primer predstavlja spekter enega vzorca. Atributi ustrezajo frekvencam oz. valovnim številom, njihove vrednosti pa absorbirani energiji pri tej frekvenci. Primeri so klasificirani v razrede, ki predstavljajo različne kemijske sestave vzorca oz. različne koncentracije določenih molekul v vzorcu. Nabor z oznako 1 na primer klasificira primere v 4 razrede: glikogen, lipid, DNA in kolagen. Uporabili smo deset naborov podatkov, ki so predstavljeni v tabeli 3.1. Večina ima okrog 200 do 400 atributov in nekaj čez 1000 primerov. Izstopajo podatki z oznako 1, ki imajo le 234 atributov in 731 primerov. Razreda sta večinoma dva, v enem naboru štirje in v enem šest. Podatke lahko tudi



Slika 3.2: Infrardeči spekter

predstavimo z grafom. Primer spektra kaže slika 3.2. Nabor podatkov z oznako 1, kjer so primeri klasificirani v štiri razrede, pa je na sliki 3.3. Na grafu so spektri za vse primere obarvani glede na razred, v katerega spadajo. V tem naboru podatkov imajo spektri posameznega razreda precej očitne značilnosti, saj lahko na grafu vidimo, kje spektri določenega razreda izstopajo. V večini naborov podatkov to ni tako očitno, saj se spektri različnih razredov bolj prekrivajo.



Slika 3.3: Nabor podatkov z oznako 1

Tabela 3.1: Nabori podatkov

številka vzorca	**število primerov**	**število atributov**	**število razredov**	**porazdelitev razredov**
1	731	234	4	214:212:195:110
2	458	1133	6	96:94:73:69:65:61
3	479	1607	2	209:189
4	199	1099	2	100:99
5	178	1099	2	101:75
6	251	1099	2	150:101
7	207	1099	2	110:100
8	317	1099	2	206:111
9	200	1099	2	100:100
10	200	1099	2	100:100

Poglavje 4

Postopek in rezultati

Za uporabo nevronskih mrež pri analizi spektrov smo morali najprej poiskati primerno arhitekturo mreže. Ker se je izkazalo, da večja globina ne prispeva k točnosti, oziroma jo celo poslabša, smo v nadaljnjih poskusih uporabljali nevronske mreže z enim in dvema skritima nivojema. Primerjali smo jih s SVM in multinomsko logistično regresijo. Da bi preverili, ali umetne nevronske mreže dobro delujejo tudi na težjih naborih podatkov, smo v zadnjem delu poskusov iz spektrov odstranili najbolj informativni del in preverili uspešnost različnih metod na preostalem delu.

4.1 Izbira arhitekture nevronskih mrež

Izbire arhitekture umetne nevronske mreže smo se lotili eksperimentalno. Ker so za različne podatke uspešne različne arhitekture mrež, smo želeli ugotoviti, ali v primeru teh spektrov obstajajo kakšne zakonitosti za izbiro arhitekture. Te poskuse smo izvajali na treh naborih podatkov z oznako 1, 2 in 3.

Najprej smo želeli ugotoviti, kako na točnost vpliva število nevronov. Spreminjali smo število nevronov na enem skritem nivoju. Število nevronov smo določili v odvisnosti od števila atributov, ki jih ima določen nabor podatkov. Točnost smo merili s CA, saj je distribucija razredov enakomerna. Število nevronov smo spreminjali od četrtine do štirikratnika števila atri-

Tabela 4.1: CA za različno globoke umetne nevronske mreže

	Število skritih nivojev			
vzorec	**1**	**2**	**3**	**4**
1	1.0000	0.9955	0.9843	0.9955
2	0.8986	0.9058	0.9058	0.9058
3	0.9373	0.9028	0.9028	0.8611

butov za faktor 2. Optimalne parametre smo izbirali s 3-kratnim internim prečnim preverjanjem, nastavljali pa smo stopnjo učenja, dropout in stopnjo regularizacije. Izkazalo se je, da spreminjanje števila nevronov sicer vpliva na točnost, vendar ne bistveno, interval števil nevronov, na katerem dobimo skoraj enako visoko in hkrati najvišjo točnost, pa je precej velik. Iz rezultatov ne bi mogli določiti splošne funkcije, kako se točnost spreminja v odvisnosti od števila nevronov, s povečevanjem števila nevronov pa ne dobimo nujno večje točnosti.

Poleg števila nevronov nas je zanimalo tudi, kako na točnost vpliva število skritih nivojev. Zgradili smo modele z 1, 2, 3 in 4 skritimi nivoji (tabela 4.1). Izkazalo se je, da večje število skritih nivojev ne prinese večje točnosti, čas gradnje modela pa je bistveno daljši. Iz teh poskusov sicer ne moremo dokončno sklepati, da globlje nevronske mreže niso smiselne, saj nam časovne in računske omejitve niso omogočale preiskati celotnega prostora parametrov. Opisani poskusi pa vseeno nakazujejo, da za naše podatke zadoščajo mreže z enim samim skritim nivojem. Zato smo pri nadaljnjih napovedih uporabljali mreže z enim in dvema skritima nivojema.

Za pomembno pa se je izkazala izbira parametrov stopnja regularizacije, dropout in stopnja učenja, ki zelo vplivajo na točnost. Nekaj primerov doseženih CA za različne izbire parametrov za vzorec 1 je predstavljenih v tabeli 4.2. Zato smo v nadaljnjih poskusih s pomočjo internega prečnega preverjanja določali optimalni nabor parametrov, čeprav se je s tem bistveno podaljšal čas gradnje modela.

Tabela 4.2: CA za različne kombinacije parametrov za vzorec 1

λ	δ	α	**CA**
1	0.5	0.1	0.2505
1	0.5	0.001	0.9746
1	0.5	0.0001	0.9354
1	0.8	0.001	0.9471
0.1	0.8	1	0.2525
0.1	0.8	0.1	0.4572
0.1	0.8	0.001	0.9804
0.1	0.8	0.0001	0.9569
0.001	0.5	0.1	0.2643
0.001	1	0.1	0.2662
0.001	1	0.001	0.9785
0.001	1	0.0001	0.9589

4.2 Uporaba različnih algoritmov

Zgradili smo modele za vseh 10 naborov podatkov z več algoritmi. Uporabili smo dve umetni nevronski mreži. Prva, ki je v tabeli z rezultati označena z *ANN 1*, ima en skriti nivo, število nevronov pa je določeno v odvisnosti od števila atributov, in sicer jih je $\frac{\text{število atributov}}{2}$. Druga z oznako *ANN 2* pa ima dva skrita nivoja. Na prvem ima št. $\frac{\text{število atributov}}{2}$, na drugem pa $\frac{\text{število atributov}}{4}$ nevronov. Pri obeh smo za aktivacijsko funkcijo izbrali ReLU. Parametre smo izbirali s 3-kratnim internim prečnim preverjanjem, pri katerem smo izbirali tri parametre. Stopnja regularizacije je označena z λ, stopnja učenja z α, dropout pa z δ. Preizkusili smo točnost za nabor parametrov, ki je predstavljen v tabeli 4.3.

Naslednji algoritem je konvolucijska nevronska mreža, ki je za infrardeče spektre smiselna, saj so sosednji atributi med sabo odvisni in si jih lahko predstavljamo tudi kot enodimenzionalne slike. Ker ima algoritem veliko časovno zahtevnost, smo uporabili en konvolucijski nivo z jedrom velikosti

Tabela 4.3: Parametri in vrednosti za interno prečno preverjanje

algoritem	parameter	nabor vrednosti
ANN 1	λ	1, 0.1, 0.01, 0.001, 0
	δ	0.5, 0.8, 1
	α	1, 0.1, 0.01, 0.001, 0.0001
ANN 2	λ	1, 0.1, 0.01, 0.001, 0
	δ	0.5, 0.8, 1
	α	1, 0.1, 0.01, 0.001, 0.0001
CNN	λ	1, 0.1, 0.01, 0.001, 0
	δ	0.5, 0.8, 1
	α	1, 0.1, 0.01, 0.001, 0.0001
SVM	λ	100, 10, 5, 1, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001
	γ	10, 5, 1, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001
MLR	λ	100, 10, 5, 1, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001

1×5. Drugi nivo je polno povezan, aktivacijska funkcija pa je ReLU. S 3-kratnim internim prečnim preverjanjem smo izbirali med enakimi parametri kot za umetne nevronske mreže (tabela 4.3).

Za primerjavo nevronskim mrežam smo zgradili modele še z dvema algoritmoma. Prvi je SVM. Uporabili smo RBF jedro, saj je to bolj primerljivo z nevronskimi mrežami kot linearno jedro. Parametra λ in γ regulirata stopnjo prilagajanja učni množici. Parametre smo izbrali s 3-kratnim internim prečnim preverjanjem (tabela 4.3).

Zadnji algoritem je multinomska logistična regresija. Multinomsko smo uporabili zato, ker imajo nekateri nabori podatkov več kot dva razreda. S 3-kratnim internim prečnim preverjanjem smo izbrali stopnjo regularizacije λ (tabela 4.3).

Tabela 4.4: CA za vse algoritme

vzorec	**ANN 1**	**ANN 2**	**CNN**	**SVM**	**MLR**	**velikost večinskega razreda**
1	**0.9909**	0.9955	0.9955	0.9824	0.8545	29,27%
2	**0.9348**	0.9219	0.8913	0.9058	0.4710	20,96%
3	0.9444	**0.9654**	0.8125	0.9167	0.7292	43,63%
4	**1.0000**	**1.0000**	0.9500	0.9500	0.6500	50,25%
5	**0.9623**	**0.9623**	0.9000	0.9622	0.8867	56,74%
6	**1.0000**	0.9868	0.9605	0.9605	0.6447	59,76%
7	**0.8571**	0.8254	0.7687	0.7460	0.5238	53,14%
8	**0.9688**	**0.9688**	**0.9688**	0.9687	0.6875	64,98%
9	**0.9833**	0.9667	0.7500	0.9166	0.5666	50,00%
10	0.9167	**0.9667**	0.7167	0.8833	0.7000	50,00%

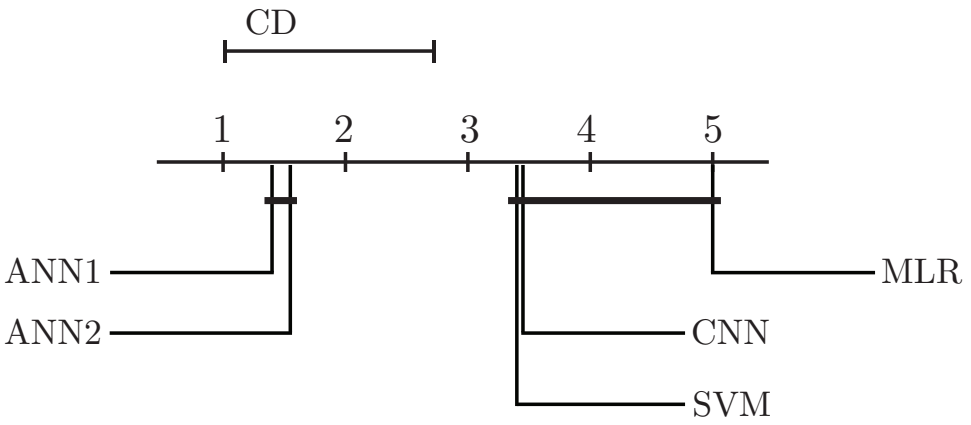
Iz tabele 4.4 lahko vidimo, da najvišje točnosti dosegajo umetne nevronske mreže, vendar to še ne pomeni, da je ta algoritem res najboljši. S statističnimi testi lahko ugotovimo, ali so opažene razlike med uspešnostjo algoritmov dovolj velike, da jih ne moremo pripisati naključju. Algoritme smo primerjali s Friedmanovim testom. Izračunani povprečni rangi (tabela 4.6) se med seboj statistično značilno razlikujejo ($\chi^2 = 34.06$, $p < 0.001$). Ker primerjamo več algoritmov, smo uporabili Nemenyijev test. Rezultati testa so prikazani na grafu. Kritična razlika je označena s *CD*. Njena vrednost pri $\alpha = 0.1$ je 1.7388 (slika 4.1), pri $\alpha = 0.05$ pa 1.9290 (slika 4.2). Algoritmi, ki niso signifikantno različni, so povezani. To pomeni, da razlike med njihovimi rangi niso dovolj velike, da bi bila primerjava med njimi statistično pravilna in zanje ne moremo določiti, kateri od njih je boljši.

Tabela 4.5: Najboljši parametri za ANN 1, ANN 2 in CNN

ANN 1			
vzorec	λ	δ	α
1	0.1	0.8	0.001
2	0	0.5	0.001
3	0.001	1	0.001
4	0.001	0.8	0.001
5	0.001	0.5	0.001
6	0.001	0.8	0.001
7	0	0.5	0.001
8	0.001	1	0.001
9	0	1	0.001
10	0	0.8	0.001
ANN 2			
vzorec	λ	δ	α
1	1	0.8	0.001
2	0	0.5	0.001
3	0.1	0.5	0.001
4	0.001	0.8	0.001
5	0.001	0.5	0.0001
6	0	0.8	0.001
7	0	0.8	0.0001
8	0.01	1	0.0001
9	0.01	0.8	0.001
10	0	0.5	0.001
CNN			
vzorec	λ	δ	α
1	0.1	0.8	0.001
2	0.1	1	0.001
3	0.01	1	0.0001
4	0.001	1	0.001
5	0.001	1	0.001
6	0.1	1	0.001
7	0.01	1	0.001
8	0.01	1	0.001
10	0	1	0.0001

Tabela 4.6: Povprečni rangi za Friedmanov test

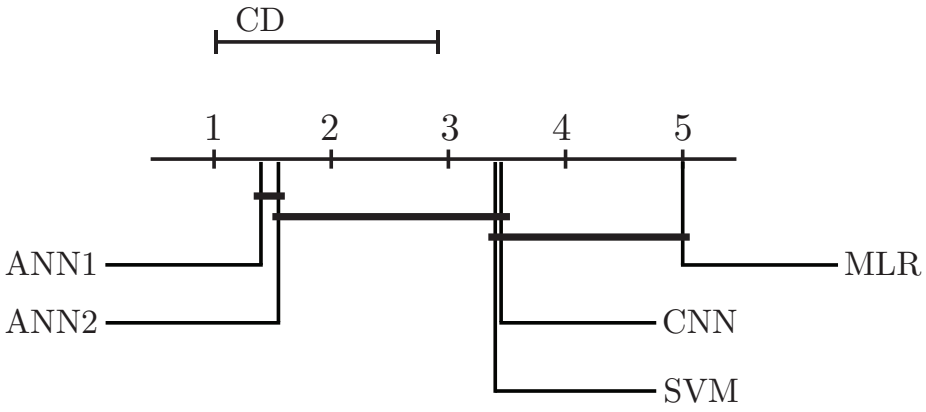
ANN 1	**ANN 2**	**CNN**	**SVM**	**MLR**
1.4	1.55	3.45	3.4	5



Slika 4.1: Diagram Friedmanovega testa za $\alpha = 0.1$

Pri manj strogem pragu $\alpha = 0.1$ sta umetni nevronski mreži statistično značilno boljši od ostalih metod, razlika med njima pa ni značilna. Tudi razlike med SVM, konvolucijsko nevronsko mrežo in multinomsko logistično regresijo so premajhne, da bi bilo pravilno trditi, kateri algoritem je boljši. Razlika med umetnima nevronskima mrežama in ostalimi algoritmi je statistično signifikantna, torej sta umetni nevronski mreži boljši od SVM, konvolucijske nevronske mreže in multinomske logistične regresije. Pri strožji meji $\alpha = 0.05$ razlika med umetno nevronsko mrežo 2 ter SVM in konvolucijsko nevronsko mrežo ni več statistično značilna. V tem primeru lahko trdimo, da je umetna nevronska mreža 1 boljša od konvolucijske nevronske mreže in multinomske logistične regresije, umetna nevronska mreža 2 pa boljša od multinomske logistične regresije. Ostalih algoritmov ne moremo primerjati.

Iz tabele 4.4 lahko vidimo, da so klasifikacijske točnosti, ki jih dosegajo



Slika 4.2: Diagram Friedmanovega testa za $\alpha = 0.05$

umetne nevronske mreže relativno visoke. Vse so višje od 0.9, od tega za 6 naborov višje od 0.95. Za konvolucijske nevronske mreže smo pričakovali boljše točnosti. Za rezultate ne moremo določiti splošnih zakonitosti. V nekaterih primerih je točnost enaka kot za umetne nevronske mreže, v drugih pa bistveno nižja. Zato so se konvolucijske nevronske mreže izkazale za nezanesljiv algoritem za klasifikacijo infrardečih spektrov. Multinomska logistična regresija je po pričakovanjih napovedovala z nizko točnostjo. SVM se je sicer s Friedmanovim testom v enem primeru izkazal za slabšega od obeh umetnih nevronskih mrež in v drugem le od umetne nevronske mreže 1. Razlike so statistično signifikantne, vendar v resnici niso tako velike, kot prikaže Friedmanov test. Pri testu namreč ni pomembno, kako velike so razlike v točnosti, ampak le, kateri algoritem napove z večjo točnostjo. Zato v primeru, ko je CA za SVM 0.9687 in za umetno nevronsko mrežo 0.9688, za SVM določimo nižji rang kot za umetno nevronsko mrežo, enako pa bi se zgodilo tudi, če bi bila razlika v točnostih na primer 0.05, kar je bistveno več. Če upoštevamo to pomanjkljivost, ne moremo reči, da je SVM veliko slabši od umetnih nevronskih mrež, saj so njegove napovedi v nekaterih primerih skoraj enako visoke kot napovedi umetnih nevronskih mrež. Glede na to

lahko sklepamo, da bomo z uporabo umetnih nevronskih mrež dosegli večjo točnost kot s SVM, ni pa nujno, da bo ta razlika zelo velika. V primerih, ko potrebujemo hiter algoritem, je SVM verjetno boljša izbira, saj je veliko hitrejši od umetnih nevronskih mrež. Za SVM traja prečno preverjanje in gradnja modela le nekaj minut, za umetne nevronske mreže pa približno tri ure. Razmerje med porabljenim časom in CA je torej za SVM boljše kot za umetne nevronske mreže.

V tebeli 4.5 so predstavljeni najboljši parametri za umetne nevronske mreže, ki smo jih izbrali z internim prečnim preverjanjem. Izkazalo se je, da vseeno obstajajo neke značilnosti za najboljše parametre. Za stopnjo učenja α je bila vedno izbrana 0.001 ali 0.0001. Za stopnjo regularizacije λ in dropout δ pa so izbrane različne vrednosti, razen pri konvolucijski nevronski mreži, kjer je δ v vseh razen v enem primeru enaka 1.

Za umetne nevronske mreže je možnih še veliko izboljšav, s katerimi bi mogoče dosegli višjo klasifikacijsko točnost. Arhitekturo smo izbrali s poskušanjem, možno pa je, da bi z drugačnimi pristopi našli boljšo. Lahko bi se tega lotili rezanjem ali večanjem mreže. Tudi konvolucijsko nevronsko mrežo bi lahko izboljšali, saj smo uporabili le en konvolucijski nivo. Morda bi se točnost povečala z drugačno arhitekturo, npr. z uporabo večjega števila konvolucijskih nivojev ali drugega jedra. Nismo imeli veliko možnosti za iskanje primerne arhitekture, saj je bil pri konvolucijskih mrežah problem v tem, da interno prečno preverjanje in gradnja modela trajata dolgo. Čas je sicer zelo odvisen od števila atributov, za večino naborov podatkov pa je to približno 20 ur. Čas bi se dalo zmanjšati tudi z uporabo združevanja maksimalnih vrednosti ali boljše opreme.

4.3 Izpuščanje dela infrardečih spektrov

Zanimalo nas je, kako bi se napovedi algoritmov spremenile, če bi ključni del spektrov oz. najbolj informativne atribute odstranili iz podatkov. Če upoštevamo te atribute, je že iz grafa razvidno, kateremu razredu pripada

spekter. Zato smo ugotavljali, kateri algoritem je primeren, kadar človek ne vidi razreda že na prvi pogled. Umetne nevronske mreže naj bi imele sposobnost, da se uspešno učijo tudi iz atributov, ki niso tako informativni, zato smo pričakovali, da bomo s poskusom pokazali, da so nevronske mreže v tem primeru boljše.

Če cel nabor podatkov predstavimo na grafu, se v idealnem primeru iz grafa vidi, kateri atributi so domnevno ključni za razlikovanje med razredi. To so tisti, pri katerih se spektri različnih razredov najbolj razlikujejo. To lahko v primeru podatkov z dvema razredoma vidimo tudi, če pogledamo vrednosti parametrov θ, ki jih dobimo z logistično regresijo. Vsak parameter pripada enemu atributu. Tisti parametri, ki so po absolutnih vrednostih največji, so koeficienti pred atributi, ki so najbolj pomembni za razlikovanje med razredi.

Uporabili smo nabore podatkov z oznako 1, 5 in 10. Te smo izbrali zato, ker je iz njihovih grafov na pogled razvidno, kje so spektri različni, v nekaterih drugih vzorcih pa je to manj očitno. Najprej smo uporabili nabor podatkov z oznako 1. Ker so podatki klasificirani v štiri razrede, smo jih morali prilagoditi. Uporabili smo tehniko eden proti vsem, kjer smo logistično regresijo ponovili štirikrat, in vsakič primere enega razreda označili za pozitivne, vse ostale pa za negativne. Vsi spektri in parametri θ, ki smo jih dobili na ta način, so predstavljeni na sliki 4.3. Vsak razred je ene barve, črta iste barve pa prestavlja vrednosti parametrov, ki smo jih dobili z logistično regresijo, ko smo za pozitiven razred označili spektre razreda, označenega z isto barvo. Da so parametri po absolutni vrednosti večji tam, kjer se spektri razreda, ki je bil z logistično regresijo označen kot pozitivni, najbolj razlikujejo od ostalih razredov, je razvidno tudi iz grafa.

Na podlagi grafa smo odstranili tiste dele spektra, kjer so črte, ki predstavljajo parametre, zelo visoko ali zelo nizko, torej kjer imajo najvišje vrhove. To sicer ni enako za vse razrede oz. za črte različnih barv, izbrali pa smo tisti del, kjer to približno velja za vse. Odstranili smo dva dela spektrov in sicer atribute z vrednostmi med 1000-1200 in 1650-1780, kar je približno 40

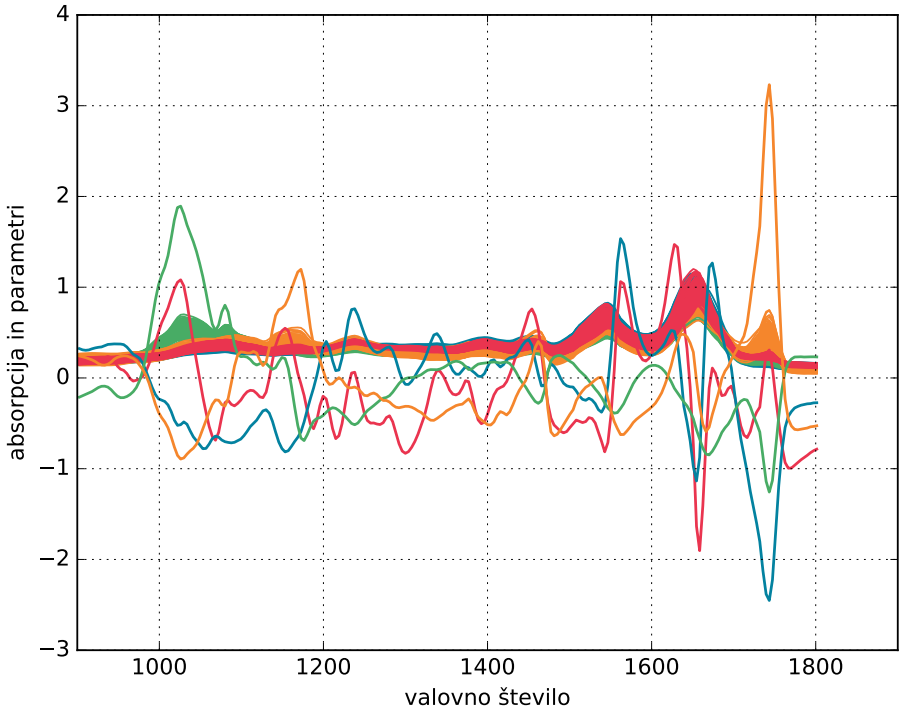
% atributov.

Razrede smo napovedali z enakimi algoritmi, kot so opisani v prejšnjem podpoglavju. Rezultati so predstavljeni v tabeli 4.7. Tu ne moremo izvesti statističnega testa, saj je velikost vzorca manjša od števila algoritmov, ki jih primerjamo. Izkazalo se je, da vse umetne nevronske mreže niso skoraj nič boljše od SVM. Vsi štirje algoritmi so napovedali s točnostjo približno 0.97. Pričakovali smo nekoliko višji rezultat od konvolucijskih mrež, saj imajo možnost, da uporabijo še dodatno informacijo o sosednjih atributih, vendar to ni prineslo izboljšave. Glede na veliko količino izpuščenih atributov tako nevronske mreže kot SVM napovedujejo dobro. Po pričakovanjih smo z multinomsko logistično regresijo dobili bistveno slabše rezultate in sicer 0.75.

Postopek smo izvedli tudi na naborih podatkov 5 in 10. Ker sta razreda samo dva, smo primere razredov označili z modro in oranžno, vrednosti parametrov pa z rdečo barvo. Pri naboru 5 smo izpustili dele spektra od 1000-1780 in 2800-3000, kar je približno 50 % atributov (slika 4.4). Najboljše se je izkazala umetna nevronska mreža 1 s točnostjo 0.9811, kar je celo več kot za celotni spekter, kjer je točnost 0.9623. Tudi za konvolucijsko nevronsko mrežo je točnost precej visoka in sicer 0.9623. Umetna nevronska mreža 2 in SVM pa imata enako točnost in sicer 0.9245. Multinomska logistična regresija pa je dosegla nizko točnost 0.7358. V tem primeru lahko rečemo, da sta najboljši mreži dobro napovedali razrede.

Pri naboru 10 pa smo izpustili 30 % atributov, in sicer 1500-1750 in 2900-3000 (slika 4.5). Umetna nevronska mreža 1 in SVM sta napovedala s točnostjo 0.9667 in 0.9666, umetna nevronska mreža 2 pa je nekoliko slabša, in sicer je točnost 0.95. S konvolucijsko mrežo smo dobili precej nižjo točnost 0.8167, z multinomsko logistično regresijo pa 0.65.

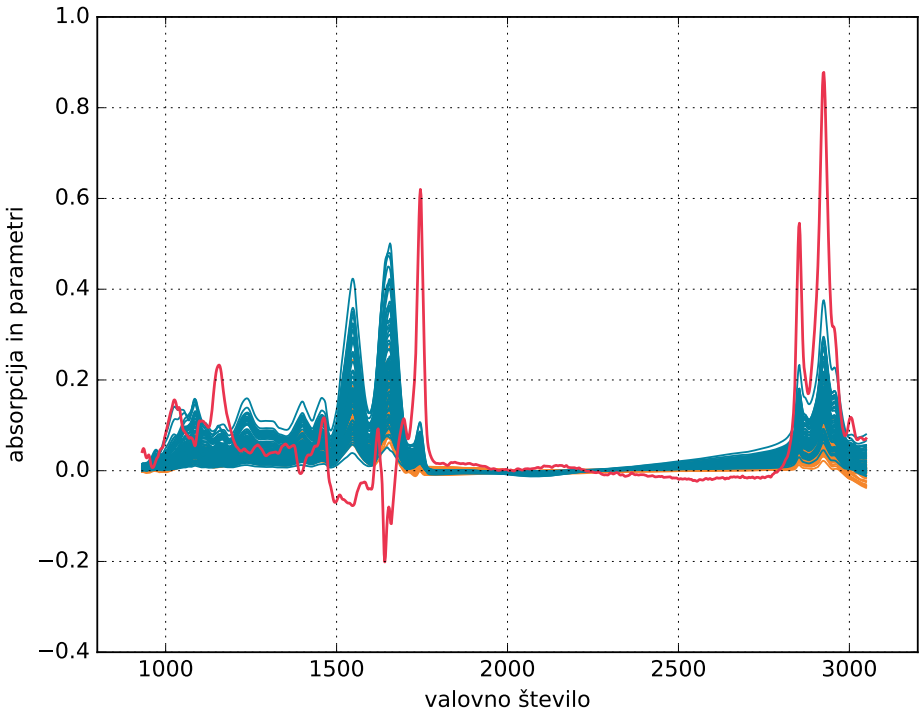
Na podlagi teh poskusov ne moremo narediti zaključka, da so umetne nevronske mreže boljše od SVM za pomanjkljive podatke. Oba algoritma pa dobro napovedujeta tudi, če upoštevamo le manj informativne atribute, saj je z njima možno doseči enako točnost, kot če bi upoštevali celotni spekter.



Slika 4.3: Spektri in parametri logistične regresije za vzorec 1

Tabela 4.7: Rezultati za vzorec 1 z manj atributi

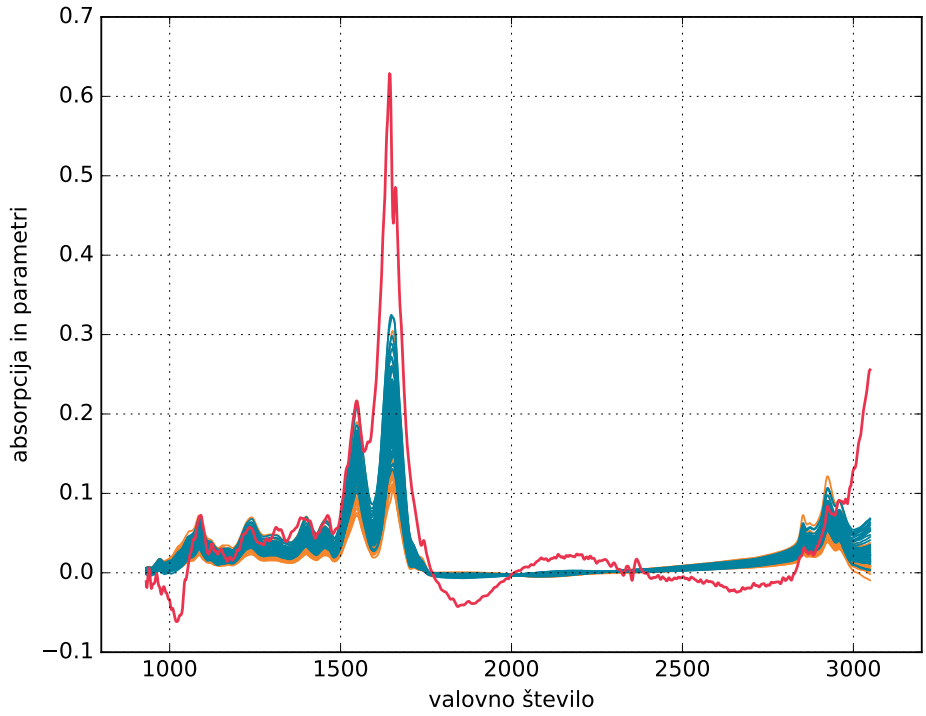
algoritem	**parametri**	**CA**
ANN 1	λ=0.1, δ=0.5, α=0.001	0.9773
ANN 2	λ=0.01, δ=0.5, α=0.001	0.9765
CNN	λ=0, δ=0.5, α=0.001	0.9785
SVM	λ=50, γ=5	0.9727
MLR	λ=1	0.7500



Slika 4.4: Spektri in parametri logistične regresije za vzorec 5

Tabela 4.8: Rezultati za vzorec 5 z manj atributi

algoritem	**parametri**	**CA**
ANN 1	λ=0, δ=1, α=0.001	0.9811
ANN 2	λ=0.1, δ=1, α=0.001	0.9245
CNN	λ=0, δ=1, α=0.001	0.9623
SVM	λ=100, γ=1	0.9245
MLR	λ=1	0.7358



Slika 4.5: Spektri in parametri logistične regresije za vzorec 10

Tabela 4.9: Rezultati za vzorec 10 z manj atributi

algoritem	**parametri**	**CA**
ANN 1	λ=0, δ=1, α=0.001	0.9667
ANN 2	λ=0.1, δ=0.8, α=0.001	0.9500
CNN	λ=0.01, δ=0.8, α=0.001	0.8167
SVM	λ=100, γ=5	0.9666
MLR	λ=1	0.6500

Poglavje 5

Zaključek

V diplomskem delu smo preučevali, kako dobre so umetne nevronske mreže za klasifikacijo infrardečih spektrov. Izmerili smo točnost za pet algoritmov, in sicer dve umetni nevronski mreži, konvolucijsko nevronsko mrežo, SVM in logistično regresijo. Izkazalo se je, da je napovedna točnost umetnih nevronskih mrež z enim ali dvema skritima nivojema najvišja, dober pa je tudi algoritem SVM, ki ni veliko slabši od umetnih nevronskih mrež. Zato je tudi SVM primeren za klasifikacijo infrardečih spektrov, saj je časovno precej manj zahteven kot umetne nevronske mreže. Konvolucijske nevronske mreže so za nekatere nabore podatkov sicer dosegale visoke klasifikacijske točnosti, za druge pa precej nizke. V splošnem so z njimi dobljene točnosti lahko zelo različne, zato taka arhitektura konvolucijske nevronske mreže ni zanesljiva za klasifikacijo infrardečih spektrov.

Naše modele bi bilo možno še precej izboljšati. Višjo točnost bi verjetno lahko dosegli, če bi izbrali bolj primerno arhitekturo umetne nevronske mreže. Tega bi se lahko lotili z uporabo umetnih nevronskih mrež z več skritimi nivoji in s tehniko rezanja ali večanja mreže. Veliko izboljšav je možnih tudi za konvolucijsko nevronsko mrežo. Z dodajanjem konvolucijskih in polnopovezanih nivojev, z uporabo združevanja maksimalnih vrednosti in drugačnega jedra bi se morda izboljšala točnost.

Rezultati naše raziskave o uporabi umetnih nevronskih mrež za klasifika-

cijo infrardečih spektrov so torej spodbudni, njihova praktična uporaba pa bi zahtevala še precej dodatnega dela.

Literatura

[1] Sajid Anwar, Kyuyeon Hwang in Wonyong Sung. Structured pruning of deep convolutional neural networks. Seoul National University.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] Convolutional neural network. Dosegljivo: `http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/`. [Dostopano 2. 8. 2016].

[4] Igor Kononenko in Marko Šikonja Robnik. *Inteligentni sistemi*. Založba FE in FRI, 2010.

[5] Steve Lawrence, Lee Giles in Ah Chung Tsoi. What size neural network gives optimal generalization? convergence properties of backpropagation. University of Maryland, 1996.

[6] Yann LeCun, Yoshua Bengio in Geoffrey Hinton. Deep learning. *Nature*, 521(444), 2015.

[7] S. Mobilio in A. Balerna. Introduction to the main properties of synchrotron radiation. Dosegljivo: `http://server2.phys.uniroma1.it/gr/lotus/Mariani_carlo/didattica/cap_01_Mobilio.pdf`. [Dostopano 20. 8. 2016].

[8] Andrew Ng. Cs294a lecture notes. Dosegljivo: https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf. [Dostopano 20. 7. 2016].

[9] Frank S. Parker. *Applications of Infrared Spectroscopy in Biochemistry, Biology and Medicine*. Plenum Press, 1971.

[10] Raul Rojas. *Neural Networks - A Systematic Introduction*. Springer, 1996.

[11] Softmax regression. Dosegljivo: http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/. [Dostopano 5. 8. 2016].

[12] Nitish Srivasta, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever in Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Communications of the ACM*, 52(8), 2014.

[13] Synchrotron. Dosegljivo: https://en.wikipedia.org/wiki/Synchrotron. [Dostopano 20. 8. 2016].

[14] Richard D. De Veaux, Richard D. De, Veaux Lyle in H. Ungar. A brief introduction to neural networks. Dosegljivo: https://people.orie.cornell.edu/davidr/or474/deveaux.pdf. [Dostopano 2. 8. 2016].